Class 2 - Controls, Layouts and Styles

Summary:

- WPF Layouts
- Examples of use of some controls
- Styles

Note: This document is based on the tutorial available at:

https://msdn.microsoft.com/en-us/library/ms752299(v=vs.110).aspx

2.1. Navigation App

Create a new Project "WPF application" with the name DETI. Run the application and see what happens.

Modify the project so that it is an application that uses navigation between pages (and not between windows). To do this, access the `MainWindow.xaml` file, change the `Window` element to `NavigationWindow` and remove the `Grid` component. You can also change other parameters (for example, change the title to DETI and change the size to 400 x 600).

You must also modify, in the associated C # file, the `Window` object for `NavigationWindow`. Run the application now and compare the result.

2.2. Adding a Window

Create a new WPF page (in the project, select `add->new Item->Page(WPF)`) with the name DETI_Home. In the XAML file, rename it to DETI_Home.

Place this window as `Source` in the `MainWindow.xaml` file by using the following code at the end of the `NavigationWindow` block:

```
Title="DETI" Height="400" Width="600" Source="DETI_Home.xaml">
```

Likewise, add another page called DETI_Cursos which will show the disciplines associated with a specific course. Set the title of the page as DETI_Cursos.

2.3. Layouts

WPF offers several layouts to organize the contents in a window, allowing different behaviour and organization of the components when resizing them.

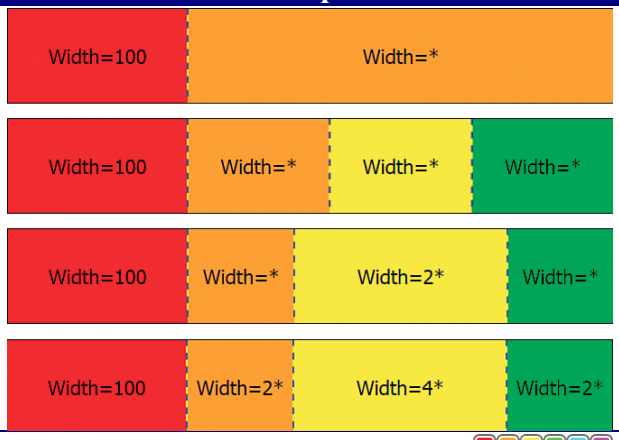
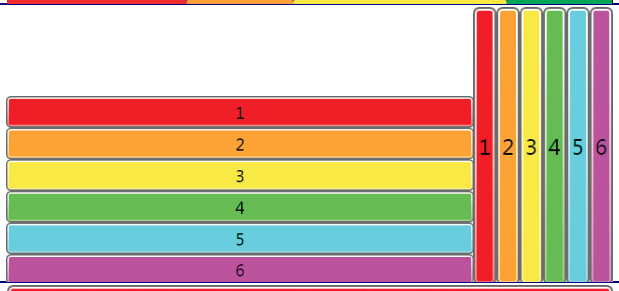
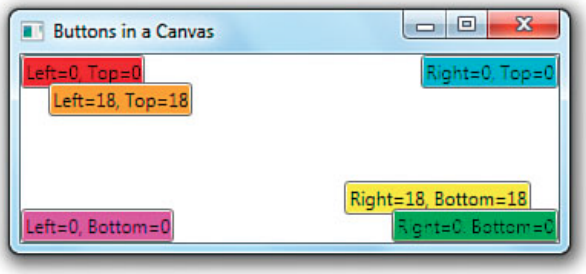
Layout	Examples	Organization
Grid		Organizes the elements in lines and columns
Stack Panel		Stacks elements horizontally or vertically
Dock Panel		Organize the elements by placing them vertically or horizontally next to each other.
Wrap Panel		Organizes the elements from left to right up to the limit of the element where it is defined.
Canvas Panel		Organizes elements by their coordinates (x, y)

Table 2-1: Main layouts available in WPF (source: WPF Unleashed - Adam Nathan)

Add a `Grid` layout with a single column and three lines in the DETI_Home window. You can define the grid using the toolbox, or by adding the following code to the xaml file. Note that the first and last lines are formatted according to the content (`Auto` option).

```
<Grid Margin="10,0,10,10">
    <Grid.ColumnDefinitions>
        <ColumnDefinition />
    </Grid.ColumnDefinitions>
    <Grid.RowDefinitions>
        <RowDefinition Height="Auto"/>
        <RowDefinition />
        <RowDefinition Height="Auto"/>
    </Grid.RowDefinitions>
</Grid>
```

Add three controls on the grid by using the keywords `Grid.Row` and `Grid.Column` to specify the position on the grid.

The interface must have a `Label` (with the text Cursos), a `ListBox` with the name cursosListBox (add 3 elements `<ListBoxItem>` for the 3 DETI's courses) and a `Button` to view the selected course.

The result should be similar to the following window. Place the `Label` "Cursos" in a `Border` element with the following characteristics to have the desired visual aspect:

```
<Border Grid.Column="0" Grid.Row="0" Height="35"
Padding="5" Background="#4E87D4">

</Border>
```

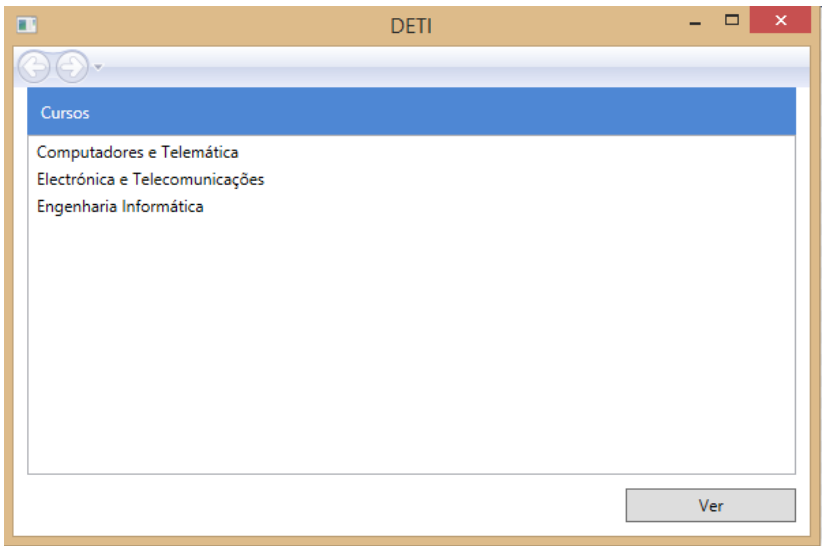


Figure 2-1: DETI_Home window interface

2.4. Adding Images

Add the provided image (Logo_UA.jpg) to the project in Visual Studio. Add another column to the left part of the window with a fixed size of 200 and add another line.

Use the `Grid.Column` and `Grid.Row` attributes to place the controls in the correct position and obtain a visual aspect like the one presented in Figure 2-1.

Add a general title by using the following code:

```
<Label Grid.Column="1" VerticalAlignment="Center"
FontFamily="Trebuchet MS"  FontWeight="Bold" FontSize="18"
Foreground="#0066cc">
    Cursos do DETI
</Label>
```

Use the provided logo to set the background image of the window by using following code:

```
<Grid.Background>
    <ImageBrush ImageSource="Logo_UA.jpg"/>
</Grid.Background>
```

The window should be similar to the picture below:

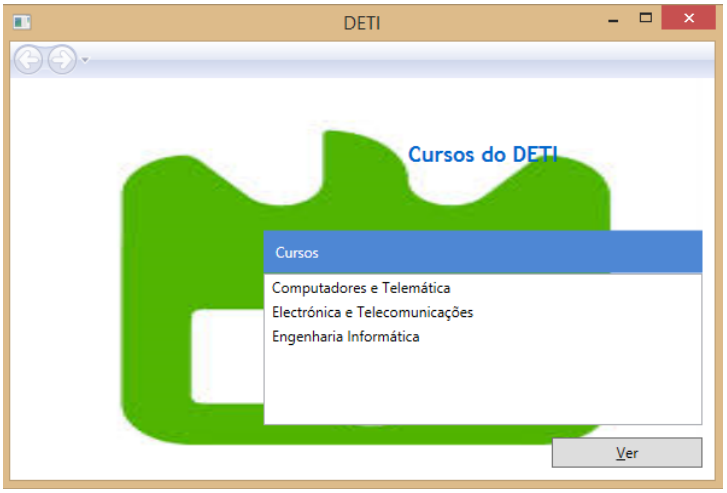


Figure 2-2: DETI_Home window interface

2.5. Navigation Between Pages

Add an event to the "Ver" button and place the following code to view another page:

```
DETI_Cursos cursosPage = new DETI_Cursos();
this.NavigationService.Navigate(cursosPage);
```

Modify the "Ver" button to enable it to be activated via the alt+v accelerator (just add an underscore before the activation letter in the name).

By reusing code as much as possible from the page created earlier, modify the DETI_Cursos page to achieve a look similar with the next window when the "Ver" button is clicked.

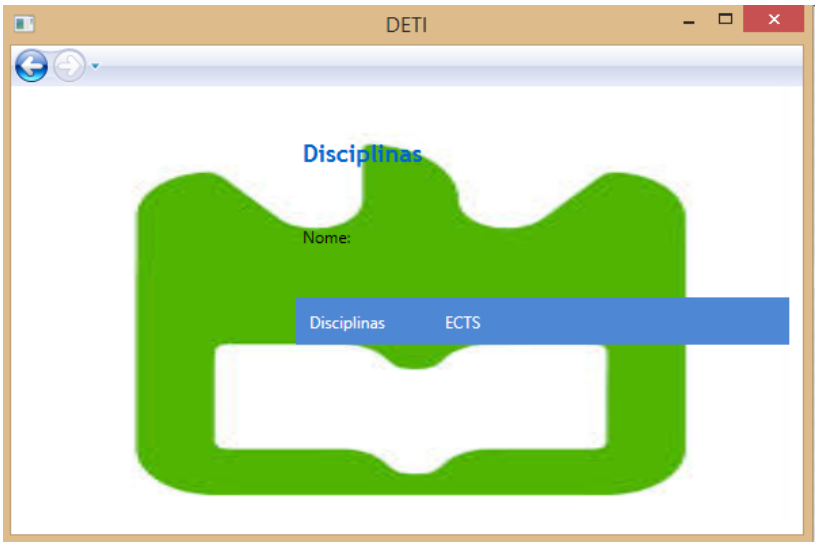


Figure 2-3: DETI_Cursos window interface

2.6. Access to Controls

Make the necessary changes so that, in front of the label "Nome" appears the name of the selected course in the `ListBox` of the DETI_Home page when the "Cursos" page is opened. You can use the following code to access and convert the selected content in a `ListBox` to `String`:

```
((ListBoxItem)cursosListBox.SelectedValue).Content.ToString()
```

Modify the code in a way that when you double click in a `ListBox` element on the home page it has the same effect as pressing the "Ver" button.

Change also the code to check if any item of the `ListBox` is selected (the item is not null) and in that case give an error message using the following code. What do the various parameters correspond to?

```
MessageBox.Show("Selecione um curso", "Erro", MessageBoxButton.OK);
```

2.7. Standalone Windows Applications

Create a new Project "WPF app" in Visual Studio in C# with the name "DETI_Windows". You can create this new project in the solution already created, thus having access to the two projects in the same solution.

Add 2 buttons with the names “Janela1” and “Janela2”. Create a new window in the previous project with the name “Exemplo1”.

In the main window code use the “Janela1” button to invoke an “Exemplo1” window with the `showDialog` method and use the “Janela2” button to invoke the same window with the `Show` method. What's the difference?

Using the following structure as a base, create a window with a `layout` similar to the example in the figure.

```
<Grid>
        <StackPanel>
              <Label />
            <WrapPanel>
                <Image Source="Logo_UA.jpg" />
                <StackPanel >
                    <Label />
                    <Label />
                </StackPanel>
            </WrapPanel>
            <Button/>
        </StackPanel>
</Grid>
```

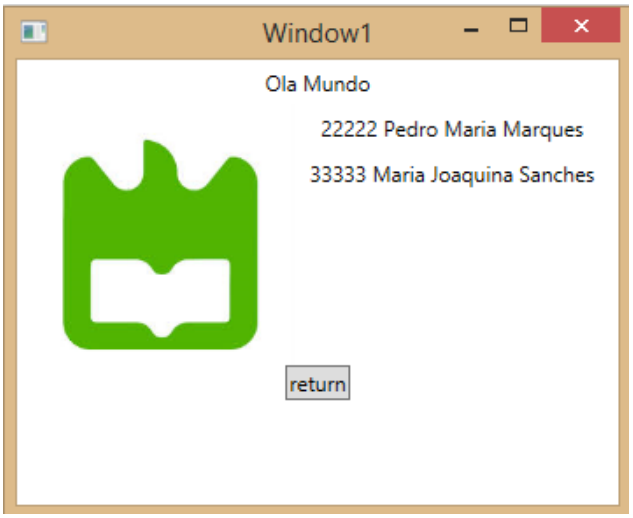


Figure 2-4: Exemplo1 window interface

Add the code needed to close the window by clicking on the return button.

Change the `WrapPanel` layout to other layouts and see the final result.

2.8. Examples of Styles and Triggers

WPF provides styles that allow you to modify the appearance or behaviour of a control. Imagine for example that you want to change the appearance and behaviour of the labels in the previous example.

Define a style in the `Application Resources` (so that it is available for the entire application) using the following code:

```
<Style x:Key="LabelStyle" TargetType="Label">
     <Setter Property="Height" Value="25" />
     <Setter Property="Width" Value="180" />
     <Setter Property="HorizontalContentAlignment" Value="Center" />
</Style>
```

Modify your program so that all `Labels` in your application use this style (use the code: `Style="{StaticResource LabelStyle}"`). Note that if you define local properties for a control, they will take precedence over the style.

Change the style to define a `Backgound LightCoral` and see the result. You can also change other properties of the `Label`.

It is also possible in WPF to use `Triggers` to modify the behavior of a control depending on conditions associated with properties, events, data changes or even the logical combination of several `Triggers`. Add the following block of code in the style of the `Labels` and observe the result.

```
<Style.Triggers>
     <Trigger Property="IsMouseOver" Value="True">
         <Setter Property="Background" Value="Red"/>
     </Trigger>
</Style.Triggers>
```